边缘计算中具有 QoS 保证的在线能耗感知任务分派

袁昊，郭得科，唐国明，罗来龙
（国防科技大学系统工程学院，湖南 长沙 410073）

摘 要：通过在网络边缘布置大量的边缘服务器，边缘计算能够为用户提供低时延、高带宽的服务。然而，大量布置边缘服务器也带来了高能耗等问题。当用户将任务从终端设备分派到不同的边缘服务器时，边缘服务器的异构性，会产生不同的能耗和时延。因此，如何在众多边缘服务器中选择一个最优的服务器进行任务分派，使得能耗和时延都比较低是具有挑战性的。提出了一种基于在线学习的具有服务质量（QoS, quality of service）保证的能耗感知任务分派方法，它可以通过与环境进行交互来获取实时的信息，从而在分派任务时，在保证 QoS 可接受的基础上，总体能耗最低。实验结果表明，与其他方法相比，提出的方法可以高效地将任务分派到最优的边缘服务器上，显著降低边缘计算网络的整体能耗。

关键词：边缘计算；任务分派；QoS；能耗感知；在线学习
中图分类号：TP3
文献标识码：A
doi: 10.11959/j.issn.2096−3750.2021.00230

Online energy-aware task dispatching with QoS guarantee in edge computing

YUAN Hao, GUO Deke, TANG Guoming, LUO Lailong
College of Systems Engineering, National University of Defense Technology, Changsha 410073, China

Abstract: Edge computing can provide users with low-latency and high-bandwidth services by deploying many edge servers at the network edge. However, a large number of deployments also bring problems of high energy consumption. When dispatching tasks from end devices to different edge servers, different energy consumption and delays will occur due to the edge servers' heterogeneity. Therefore, it is a challenge to select an optimal server among many edge servers for task dispatching so that energy consumption and delay are relatively low. An energy-aware task dispatching method with quality of service (QoS) guarantee based on online learning was proposed. It can obtain real-time information by interacting with the environment to ensure energy consumption was minimal while the QoS was acceptable when dispatching tasks. Experiments show that the proposed method can dispatch tasks efficiently to the optimal server compared with other methods, thereby reducing the edge computing network's overall energy consumption significantly.

Key words: edge computing, task dispatching, QoS, energy-aware, online learning

1 引言

物联网[1]的发展显著地提升了人们的生活质量，人们可以通过将终端设备（如智能手机、智能电视、智能手表等）接入物联网实现“万物互联”，从而提高生活的便利性。然而，随着用户终端处理的任务越来越多，受限于终端设备有限的算力和电池的容量，终端设备很难应对大量计算密集型任务。云计算[2]的出现很好地解决了终端设备算力不足和电池寿命有限等问题，用户可以将计算密集型

收稿日期：2021−01−11；**修回日期**：2021−03−01
通信作者：郭得科，guodeke@gmail.com
基金项目：国家自然科学基金资助项目（No.U19B2024）
Foundation Item: The National Natural Science Foundation of China (No. U19B2024)

的任务卸载到远程的云计算中心上来进行计算，从而提升计算的速度并且可以延长终端设备的使用寿命。然而，由于云计算中心通常距离用户较远，这会带来较大的传输时延，从而导致 QoS[3]较低，这对于一些时延敏感性高的任务（如实时视频处理）来说是不可接受的。

边缘计算[4]的出现弥补了以往云计算中心距离用户较远所带来的高时延、低带宽等不足。不同于云服务器，边缘计算通过在网络边缘布置大量的边缘服务器，使得用户可以将任务从终端设备上卸载到附近的边缘服务器上进行计算，以此来减轻计算密集型的任务对终端设备带来的计算压力和能耗负担，并延长终端设备电池的寿命。相比于云服务器充足的计算资源，边缘服务器的计算资源相对较少。此外，由于边缘服务器布置的范围较广且数量较多，其产生的能耗也会非常大。因此，将任务分派到哪个边缘服务器上进行运算才能使得时延较低，同时保证能耗较低就显得尤为重要。

事实上，如何将任务分派到最优的服务器上，使得时延和能耗均较低面临着诸多挑战。首先，任务的到达是随机的，增加了任务分派的不确定性，传统离线优化方法的效果较差。其次，服务器的能耗也会随着被分派到服务器上的任务数量的变化而变化。另外，随着物联网设备产生的任务数量的增加，其求解空间也会非常大，传统动态节能的任务分派策略的求解也会非常难。因此，设计一种高效且复杂度较低的在线分派策略，使得时延和能耗较低是很有必要的。

然而，传统的分派方法要么追求较低的时延[5]，要么追求较低的能耗[6]，很难做到同时兼顾时延和能耗。低时延和低能耗是两个很重要的指标，需要同时考虑。在本文中，提出一种基于在线学习（OL, online learning）[7]的边缘计算任务分派方法。它通过与环境进行交互来实时地获取服务器负载等信息，并根据当前任务和服务器的状态动态地将任务分派到最优的服务器上，在保证时延可接受的基础上使得系统的总能耗最小化。同时，它还可以通过不断地交互来提升算法的决策能力，以此来应对任务到达的随机性和服务器负载的动态性。通过在真实数据集上进行实验，验证本文提出的方法的优越性。结果显示，与其他方法相比，本文提出的方法可以在保证时延可接受的基础上，使得服务器产生的能耗最小，显示出了本文提出的算法的高效性。

2 相关工作

随着边缘计算的发展，边缘计算中任务分派的方法也被广泛地研究。通常来说，任务分派策略常被分为两个类别，一类是基于性能的分派策略，另一类是基于能耗的分派策略。

基于性能的分派策略的目标是最小化任务的平均完成时间。边缘计算中最早提出的任务分派和调度方法是 Ondisc[8]，它的基本思想是将任务分派到带来的总权重响应时间最短的边缘服务器上。Jia 等[9]提出一种基于排队论和启发式策略的分派方法，它可以达到负载均衡的效果。Meng 等[10]提出一种在线任务分派和调度方法——Dedas，它将任务分派到通过调度方法计算后产生的完成任务的数量最大和完成任务的时间最短的边缘服务器上。

基于能耗的分派策略的目标是降低系统的能量消耗。Huang 等[11]提出一种利用李雅普诺夫优化来进行任务的动态分配，以此来降低系统能耗的方法。Chen 等[12]提出了一种节能动态卸载算法——EEDOA，它可以在最小化系统能耗的同时保证任务队列在一定限度内。Lin 等[13]提出一种任务调度算法来降低系统的总能耗，但它没有考虑任务分派过程中的能量分配。Guo 等[14]提出一种分派方法——eDors，它通过动态地分派和调度任务从而达到能耗最优的效果。

然而，这些方法一般追求较短的时延或者较低的能耗，不能够做到两者兼顾。在本文中，提出一种基于 OL 的在线任务分派方法，它可以有效地解决上述问题。具体地，它通过与环境进行交互来获取环境的实时信息，并同时将时延和能耗两种指标考虑在内，在保证时延可接受的情况下，最小化总体的能耗，从而在众多服务器中选择最优的服务器。

3 系统模型和问题描述

在本节中，将介绍系统模型、问题描述和难点分析。

3.1 系统模型

系统架构如图 1 所示，它包括用户的终端设备、接入点（AP, access point）、边缘服务器。

1) 网络模型

图 1 包含 J 个边缘服务器 $E=\{E_1,E_2,\cdots,E_J\}$。对于每个边缘服务器，已配置了多种应用程序和服务，

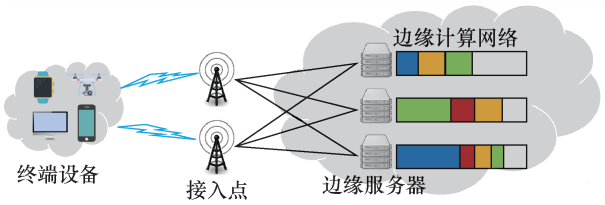


图 1　系统架构

因此可以为用户直接提供服务。除此之外有 K 个 AP，用户可以通过无线方式连接距离终端设备最近的 AP。由于终端设备与 AP 之间的传输时延很小，因此，在本文中不考虑两者之间的时延。接入点 k 与边缘网络中的边缘服务器之间的连接关系可以用如式(1)所示向量来表示。

$$\boldsymbol{A}_k=[a_{k,1},a_{k,2},\cdots,a_{k,J}] \tag{1}$$

其中，$a_{k,j}=0$ 表示接入点 k 与边缘服务器 j 之间无连接；反之，$a_{k,j}=1$ 表示接入点 k 与边缘服务器 j 之间有连接。将接入点 k 和边缘服务器 j 之间的传输时延记为 $e_{i,j}$，用户通过将任务请求从个人设备卸载到距离最近的 AP 上，然后基站通过分派方法将卸载的任务分派到最优的边缘服务器上执行运算操作。在本文中，考虑使用离散时间模型，即总时间被分割成长度为 Δt 的时间段，记为 $W=\{1,2,\cdots,T\}$。

2) 服务器模型

通常来说，边缘服务器被布置在距离 AP 很近的地方，或者被布置在 AP 上，从而便于较快地处理。在本文中，假设的服务器总数为 J 个。每个服务器的内存容量为 S_j，它是一个正值，表示服务器上可以配置有限数量的应用，且只能处理一定数量的任务。假设所有服务器上都配置了任务所需要的应用，不需要考虑从远程云服务器上来将应用配置到边缘服务器上。其次，每个服务器上的计算资源记为 C_j，表示每个时间段可以处理的数据量，单位为bit / Δt，所有被卸载到服务器上的任务均等分配计算资源。在本文中，不允许任务从一个服务器上迁移至另外一个服务器上进行处理。

3) 任务模型

任务会从用户的终端设备上随机生成。每一个任务生成后，终端设备会将其卸载至距离用户最近的 AP 上，然后 AP 会将任务分派到最优的服务器上。将任务 i 所携带的任务量大小记为 r_i，单位为 bit。任务 i 到达服务器的时间记为 a_i。当任务 i 到达服务器后，它会与其他任务均等分计算资源。每个时段服务器可以处理任务 i 的数据量大小为 $c_t^j=\dfrac{C_j}{N_t^j}$，N_t^j 为 t 时段在服务器 j 上的任务数量，则任务到达服务器后每时段的处理量可以表示为

$$\boldsymbol{C}_{i,j}=[c_a^j,c_{a+1}^j,\cdots,c_{a+p}^j] \tag{2}$$

其中，a 为任务从终端释放的时间，p 为任务的处理时长，且满足 $r_i=\sum_{t=a}^{a+p}c_t^j$。除此之外，每个任务还有最后期限记为 d。最终，将每个任务 i 的属性记为一个向量 $\boldsymbol{O}_i=(v_i,a_i,p_{i,j},r_i,d_i)$，$v_i$ 表示距离用户最近的 AP，即任务将被卸载到接入点 v_i 上。此后，AP 需要决定将任务分派到哪个服务器上才能在保证 QoS 的基础上尽可能降低能耗。

3.2　问题描述

在时段 W 中，用户从终端设备随机产生任务集 $R=\{1,2,\cdots,N\}$。首先将其卸载至距离用户最近的 AP 上，接下来，AP 通过分派方法将卸载的任务进行分派，将任务 i 的分派向量记为

$$\boldsymbol{F}_i=[f_1,f_2,\cdots,f_J] \tag{3}$$

其中，当 $f_j=1$ 表示任务将会被分派至边缘服务器 j 上进行处理，且任务只能从接入点 k 分派至与其相连接的边缘服务器上，即 $a_{k,j}=1$。

任务被分派到服务器上后，会和其他被分派的任务共享边缘服务器上的计算资源，并经过 p 时长完成计算。因此，任务 i 的响应时间为 $2e_{i,j}+p_{i,j}$。服务器负载与能耗的关系如图 2 所示。

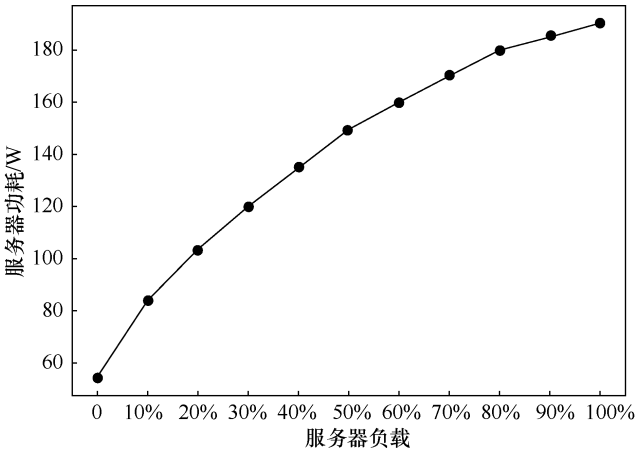


图 2　服务器负载与能耗的关系

如图 2 所示，服务器在运行过程中会产生大量能耗，且与服务器负载成正比关系。当得到当前服务器负载后，便可对应求出其能耗大小，通过对所有服务器 0～T 时段内能耗进行累加，便可获得系

统的总体能耗。具体地，在任意时段 t，边缘服务器 j 的能耗 E_t^j 可由式(4)求出。

$$E_t^j = h(l_t^j) \tag{4}$$

l_t^j 可由式(5)计算得到。

$$l_t^j = \frac{\sum r_i}{S_j}, i \in R_t^j \tag{5}$$

其中，R_t^j 为 t 时段边缘服务器 j 上的任务集，$h(\cdot)$ 的映射关系如图 2 所示。最终，优化问题可以表示为

$$\min_{F_i} \sum_{t=1}^{T}\sum_{j=1}^{J} E_t^j \tag{6}$$

$$\text{s.t. } \frac{1}{N}\sum_{i=1}^{N} 2e_{i,j} + p_{i,j} < \varepsilon \tag{7}$$

$$\sum r_i < S_j, i \in R_t^j, j \in J \tag{8}$$

其中，ε 表示用户可接受的最大平均任务响应时间，用来衡量 QoS。

3.3 难点分析

1) 任务产生的随机性

在问题建模中，假设任务集和服务器负载是已知的，因此可以用离线优化的方法进行求解。然而，这些假设在实践中是不现实的。事实上，由于卸载任务的随机性，边缘计算网络中的任务到达是随机的。此外，由于服务器负载是动态变化的，进一步增加了问题的不确定性。因此，使用离线优化的解决方案可能与实际情况相差甚远，需要一种在线的方法进行求解。

2) 计算的高复杂度

上述优化问题中嵌入了很多 NP 难子问题。首先，在现实场景中，AP 和边缘服务器的数量多至数百台甚至数千台。传统的优化方法在分派中需要花费很长时间来寻找最优任务传输链路和边缘服务器。其次，用户的终端设备会向边缘服务器发送大量的任务请求，导致复杂的分派问题。上面的两个挑战会产生一个巨大的搜索空间，因此通过任何蛮力的搜索方法来求解最优值都是低效的。

为了应对上述挑战，接下来提出了一种基于 OL 的在线任务分派方法。

4 基于 OL 的任务分派方法

在本节中，提出了一种基于多臂老虎机（MAB, multi-armed bandit）[15]的在线任务分派方法，它可以利用过去的经验（即历史路径和服务器选择）来估计当前环境的状态（例如，传输路径的实时带宽和服务器的负载），从而选择最优边缘服务器来卸载任务。

4.1 成分及概念

本节将描述基于在线学习（OL）的分派方法的具体实现细节，它使用多臂老虎机模型并具有时变奖励（reward）。算法通过不断地与环境进行交互来获取实时信息（即奖励），并更新手臂（arm）权值，从而使得在每一次决策时，都能够选择最优的手臂（即将任务分派到最优的服务器上）。

1) 手臂

在 MAB 问题中，最基本的组成部分是手臂。算法将可选择的选项视为手臂，在每次决策时，算法都要从所有手臂中选择最优的手臂，即做出最优决策。本文将边缘服务器视为 arm，$\mathcal{J} = \{1,\cdots,J\}$ 表示 J 个手臂的集合。算法可以通过 t−1 时刻之前选择的 arm 所反馈的奖励来更新手臂的权值，以此来确定 t 时刻选择哪个目标 arm。

2) 奖励

算法选择目标 arm 后，会与环境交互并获得奖励。在本文中，设计的奖励包含两部分 $u_j(t)$ 和 $v_j(t)$，它们分别是 QoS 和能耗的度量，并将其称为复合奖励。具体地，$u_j(t)$ 和 $v_j(t)$ 定义为

$$u_j(t) = 2e_{i,j} + p_{i,j} \tag{9}$$

$$v_j(t) = E_t^j \tag{10}$$

相应地，复合奖励的定义为

$$g_j(t) = \begin{cases} \alpha \exp(-v_j(t)), u_j(t) < \varepsilon \\ 0, u_j(t) \geqslant \varepsilon \end{cases} \tag{11}$$

式(11)表示只有当可选择的服务器 j 的 QoS 在用户可接受的范围内时，才会获得相应的奖励，从而便于下次决策时选择最优服务器。奖励只有当任务 i 完成处理并返回 AP 时，arm 的权重才会更新，否则，它将保持当前的权重。本文方案的目标是最大化总的复合奖励，即 $\mathrm{E}\left[\sum_{t=1}^{T} g_{j(t)}(t)\right]$，其中 $j(t)$是 t 时段选择的 arm。因此，为了使得最终的复合奖励值最大，算法会通过与环境不断交互来获取实时信息，从而在每一步都做出最优决策。

3) 悔度

为了评估算法的效率，MAB 中引入了悔度（regret）的概念。悔度是最优 arm 和被选择 arm 之间奖励的差异。令 $j^*(t) = \max_j | g_j(t), 1 \leqslant j \leqslant J$，

$j^*(t)$ 是在时间 t 时的最优 arm。本文方案的目标是设计一个策略 π 来选择 arm$j(t)$，使得策略与 Oracle 之间的悔度尽可能小。具体地说，Oracle 是一个最优策略，它知道所有网络和边缘服务器的参数，因此在每个时间 t 都可以选择最优的 arm $j^*(t)$。策略 π 的悔度被定义为

$$R_\pi(T)=\mathrm{E}\left[\sum_{t=1}^{T} g_{j^*(t)}(t)-g_{j(t)}^{\pi}(t)\right] \tag{12}$$

由于 Oracle 完全了解参数 $u_j(t)$ 和 $v_j(t)$，所以它可以一直选择最优的 arm。但是，这些参数只能在每次选择 arm$j(t)$时进行估计，无法直接获取。因此，设计策略 π 使得其能够实时估计环境状态，并以此来进行决策分析，最终最大化总复合奖励（或最小化总悔度）是具有挑战性的。基于 OL 的任务分派方法如算法 1 所示。

算法 1　基于 OL 的任务分派方法

输入　任务集 $R=\{1,2,\cdots,N\}$

输出　选择手臂 $j(t),1\leqslant t\leqslant T$

For t=1:Jdo

选择手臂 $j(t)=t$

接收手臂 j 的 $u_j(t)$、$v_j(t)$

手臂 j 的复合奖励 $g_j(t)$

更新手臂 j 的平均复合奖励 $M_j(t)$

$N_j(t)=1$

end

For t=J+1:Tdo

选择手臂 $j(t)=\arg\max_j\left[\left[M_j(t)+c\sqrt{\dfrac{\ln t}{N_j(t)}}\right]\right]$

接收手臂 j 的 $u_j(t)$ 和 $v_j(t)$

手臂 j 的复合奖励 $g_j(t)$

更新手臂 j 的平均复合奖励 $M_j(t)$

$N_j(t)=N_j(t-1)+1$

end

4.2　策略设计

在本文中，提出了一种基于置信区间上界（UCB, upper confidence bound）[16-17]的在线分派方法，它会选择有最大平均奖励值的 arm。具体来说，其奖励的本质是均值的标准差，它反映了候选 arm 的不稳定性，是置信区间的上界。在每次决策时，算法会遵循正面不确定性的乐观原则，选择具有最大置信区间上界的 arm，其定义为

$$j(t)=\arg\max_j\left[M_j(t)+c\sqrt{\frac{\ln t}{N_j(t)}}\right],1\leqslant j \tag{13}$$

$$M_j(t)=\frac{1}{t-1}\sum_{i=1}^{t-1} g_j(t),t\geqslant 2 \tag{14}$$

如算法 1 所示，$M_j(t)$ 为 arm$j$ 的 t 时段前复合奖励的均值，$N_j(t)$ 为 arm j 的选择次数，c 为控制探索和利用的参数。具体地，算法首先对每个 arm 进行一次选择，以获得每个 arm 的初始复合奖励（如算法 1 的第 1～7 行），然后选择置信区间上界最大的 arm，并在每次选择完后对每个 arm 的权值进行更新（如算法 1 的第 8～14 行）。在算法训练过程中，将奖励函数中的 α 设定为 1，式(13)中的 c 设定为 0.3。除此之外，算法已被证明其每轮计算都较为简单，即可以提供 $O(\log T)$ 的悔度保证。

5　实验评估

在本节中，将通过仿真实验并使用来自谷歌集群的大量真实数据，与对比方法（baseline）进行比较，以评估本文提出方法的性能。仿真实验是在多台个人计算机上进行的，每台计算机可以视为一个边缘服务器。计算机的 CPU 为 Inteli7-9750H 2.6 GHz，内存为 16 GB DDR4 2 666 MHz，操作系统为 64 bit Ubuntu 16.04。算法是基于 Python 3.7 开发设计的。

5.1　实验设置

使用来自谷歌集群的数据集，它包括到达时间、处理时间和截止日期等信息。数据集由 500 个任务组成，这些任务不仅包括大型任务（如大数据分析和实时视频处理），还包括小任务（如图像处理和虚拟现实）。使用真实的网络拓扑结构，随机选取 10 个点放置边缘服务器并进行仿真实验。

5.2　baseline

为了更好地评价本文方法的性能，反映其在任务分派方面的效率，将其与以下 3 种对比方法进行了对比实验。

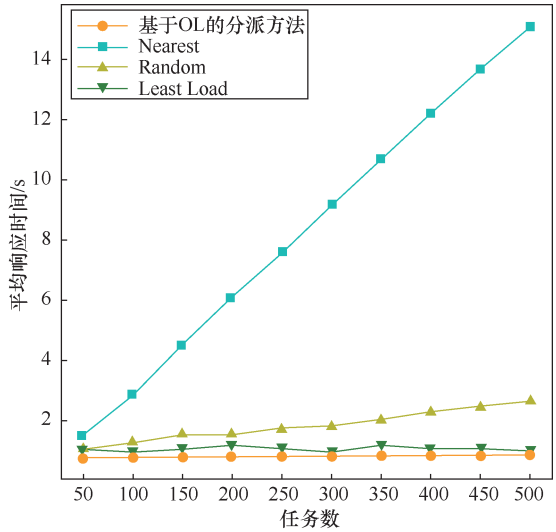
1) Nearest[18-19]：分派任务去距离 AP 最近的边缘服务器。

2) Random[5]：随机分派任务去一个边缘服务器。

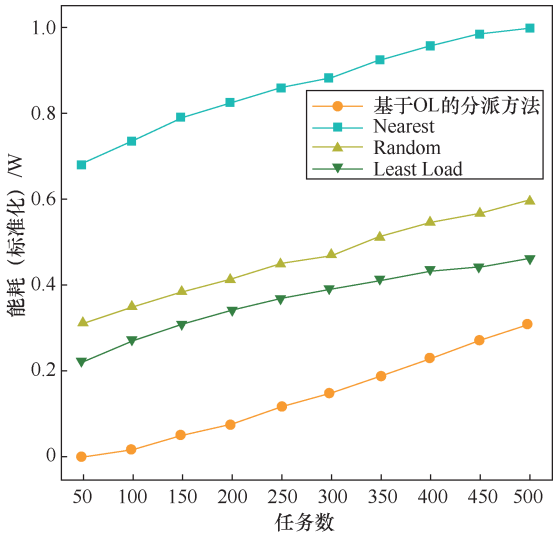
3) Least Load[5]：分派任务去等待时间最短的边缘服务器。

5.3 实验结果

任务数对任务平均响应和服务器能耗的影响如图 3 所示，从图 3(a)中可以看出，由于 Nearest 将任务分配到距离 AP 最近的服务器上，随着任务数的逐渐增多，每个任务分配的计算资源就会变少，处理效率也会随之下降，任务的平均响应时间增加，从而增加了系统的总能耗。Random 会随机地将任务分派到附近的服务器上，从而缓解任务数增加带来的负担，然而由于其算法的随机性会导致效果不稳定。Least Load和基于OL的分派方法都可以动态地将任务分派到最优的服务器上，因此当任务数逐渐增加时，任务的平均响应时间可以保持在一个比较稳定的水平。



(a) 任务数对任务平均响应的影响



(b) 任务数对服务器能耗的影响

图 3　任务数对任务平均响应和服务器能耗的影响

从图 3(b)中可以看出，由于 Nearest、Random 和 Least Load 3 种对比方法并没有考虑能耗因素，因此，即使其 QoS 可以保持在一个可接受的范围内（如 Least Load），也会造成较高的能源消耗。Nearest 在分派任务时，会导致任务的平均响应时间较长，拉长了服务器运行的时间，从而产生较高的能耗。Random 也由于算法效果的不稳定性导致其效果不佳。基于 OL 的分派方法可以在保证 QoS 的基础下选择能耗最低的服务器，因此其在 4 种分派方法中的效果最好。

6 结束语

为了解决边缘计算服务器能耗较大的问题，本文提出了一种基于 OL 的分派方法，它可以通过不断与环境进行交互来获得当前边缘服务器的状态，并在保证 QoS 可以接受的情况下，选择能耗最低的服务器。实验结果表明，与其他方法相比，本文提出的方法可以在保证 QoS 较高（即任务的平均响应时间较短）的情况下，选择能耗最低的服务器进行任务分派。

参考文献：

[1] ASHTON K. That “Internet of Things” thing[J]. RFID Journal, 2009, 22(7): 97-114.

[2] HAYES B. Cloud computing[J]. Communications of the ACM, 2008, 51(7): 9-11.

[3] CAMPBELL A, COULSON G, HUTCHISON D. A quality of service architecture[J]. ACM SIGCOMM Computer Communication Review, 1994, 24(2): 6-27.

[4] SHI W S, CAO J, ZHANG Q, et al. Edge computing: vision and challenges[J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[5] MENG J Y, TAN H S, XU C, et al. Dedas: online task dispatching and scheduling with bandwidth constraint in edge computing[C]//2019 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 2287-2295.

[6] ZHANG K, MAO Y M, LENG S P, et al. Energy-efficient offloading for mobile edge computing in 5G heterogeneous networks[J]. IEEE Access, 2016(4): 5896-5907.

[7] TERRY A, FATHI E, et al. The theory and practice of online learning[M]. New Brunswick: Athabasca University Press, 2008.

[8] HAN Z H, TAN H S, LI X Y, et al. OnDisc: online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds[J]. IEEE/ACM Transactions on Networking, 2019, 27(6): 2472-2485.

[9] JIA M K, CAO J N, LIANG W F. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2017, 5(4): 725-737.

[10] MENG J Y, TAN H S, XU C, et al. Dedas: online task dispatching and

scheduling with bandwidth constraint in edge computing[C]//2019 IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 2287-2295.

[11] HUANG D, WANG P, NIYATO D. A dynamic offloading algorithm for mobile computing[J]. IEEE Transactions on Wireless Communications, 2012, 11(6): 1991-1995.

[12] CHEN Y, ZHANG N, ZHANG Y C, et al. Energy efficient dynamic offloading in mobile edge computing for Internet of things[J]. IEEE Transactions on Cloud Computing, 2019(99): 1.

[13] LIN X, WANG Y Z, XIE Q, et al. Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment[J]. IEEE Transactions on Services Computing, 2015, 8(2): 175-186.

[14] GUO S T, XIAO B, YANG Y Y, et al. Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing[C]// Proceeding of the 35th Annual IEEE International Conference on Computer Communications. Piscataway: IEEE Press, 2016: 1-9.

[15] GITTINS J, GLAZEBROOK K, WEBER R. Multi-armed bandit allocation indices[M]. Chichester: John Wiley & Sons, 2011.

[16] GARIVIER A, MOULINES E. On upper-confidence bound policies for switching bandit problems[C]//Proceeding of Algorithmic Learning Theory. [S.l.:s.n.], 2011: 174-188.

[17] KAUFMANN E, CAPPÉ O, GARIVIER A. On Bayesian upper confidence bounds for bandit problems[C]//Proceeding of Artificial Intelligence and Statistics. [S.l.:s.n.], 2012: 592-600

[18] JIA M K, CAO J N, LIANG W F. Optimal cloudlet placement and user to cloudlet allocation in wireless metropolitan area networks[J]. IEEE Transactions on Cloud Computing, 2017, 5(4): 725-737.

[19] URGAONKAR R, WANG S Q, HE T, et al. Dynamic service migration and workload scheduling in edge-clouds[J]. Performance Evaluation, 2015, 91: 205-228.

[作者简介]

袁昊（1998− ），男，国防科技大学系统工程学院硕士生，主要研究方向为边缘计算、绿色计算等。

郭得科（1980− ），男，博士，国防科技大学教授，主要研究方向为网络计算与系统、分布式计算与系统、网络空间安全、大数据分析处理、移动计算等。

唐国明（1986− ），男，博士，国防科技大学副教授，主要研究方向为边缘计算、绿色计算等。

罗来龙（1991− ），男，博士，国防科技大学讲师，主要研究方向为计算机网络、数据结构等。